CERTIFICATE OF MAILING/TRANSMISSION
I hereby certify that this correspondence is being submitted electronically via EFS Web on the date shown below.

/Rob Brownstein/ October 21, 2008
Rob Brownstein ***October 21, 2008***

Attorney Docket No.: 42P18331 *Patent*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)	
	)	
Shah et al.	)	Examiner: Larry D. Donaghue
	)	
Application No.: 10/809,077	)	Art Unit: 2154
	)	
Filed: March 24, 2004	)	Confirmation No.: 7658
	)	
For: MESSAGE CONTEXT BASED TCP TRANSMISSION	)	
	)	
	)	

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF (37 C.F.R. § 41.37)

Sir:

This appeal brief is submitted pursuant to the Notice of Appeal filed August 22, 2008, for the above-captioned patent application. The Notice of Appeal and this Appeal Brief is filed in response to the Final Office Action mailed May 23, 2008. Appellants respectfully request consideration of this appeal and allowance of the application by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The real party in interest in this appeal is Intel Corporation ("Intel"), a Delaware corporation having a principal place of business at 2200 Mission College Blvd., Santa Clara, California, 95052. Intel is the assignee of the entire right, title and interest in the above-captioned application by virtue of an assignment recorded at the U.S. Patent Office at Reel 015466, Frame 0177.

II. RELATED APPEALS AND INTERFERENCES

Appellants and Appellants' legal representative know of no interferences, appeals, or other proceedings that will directly affect, be directly affected by, or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-13 and 17-21 are pending in the application and are the claims on appeal. All claims in the application currently stand rejected based on U.S. Patent Publication No. 2002/156927 A1 to *Boucher et al.* (hereinafter "Boucher"), U.S. Patent No. 7,017,042 to *Ziai et al.* (hereinafter "Ziai"), and allegedly Admitted Prior Art. The basis of rejection of the claims is as follows:

(i) Independent claims 1, 9, and 17 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Boucher in view of Ziai.

(ii) Dependent claims 2-5, 8, 10, 12, 13, 20, and 21 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Boucher in view of Ziai.

(iii) Dependent claims 6, 7, 18, and 19 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Boucher in view of allegedly Admitted Prior Art.

(iv) No basis for rejection of dependent claim 11 has been presented in either the Final Office Action mailed May 23, 2008 or the first Office Action mailed December 18, 2007.

IV. STATUS OF AMENDMENTS

As of the Final Office Action mailed May 23, 2008, claims 1-21 were pending in the application. Applicants filed an Amendment After Final on August 22, 2008 cancelling claims 14-16 to place the application in better form for appeal. No indication that the Amendment After Final cancelling claims 14-16 was rejected has been received by Appellants. Accordingly, it is believed the claims on appeal are claims 1-13 and 17-21 and further these claims incorporate all previous amendments.

A copy of all claims on appeal, as finally rejected on May 23, 2008, is attached hereto in Appendix A.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The specification of the instant application discloses a technique for the transmission of packets using an offload engine, such as a TCP Offload Engine (TOE), to at least partially relieve the burden of network protocol operations from a host processor (e.g., see para. [0027]). For example, the network protocol operations performed in the offload engine and relieved from the host processor may include protocol processing associated with the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Protocol (IP), or application layer protocols (e.g., see para. [0027]). The protocol processing for a packet is performed by the offload engine while the packet payload waits in the host memory (e.g., see para. [0024]). The technique avoids the overheads associated with a store-and-forward stage in the TCP path of conventional offload engines by directly transmitting message buffers or packet payloads from host memory without intermediate buffering of the packet payloads in the offload engine (e.g., see para. [0019]).

Embodiments of the present invention, as recited in independent claims 1, 9, and 17, relate generally to a method, apparatus, or machine-readable medium providing instructions for transmitting packets. To transmit packets a protocol control block (e.g., see TCB 122 in FIG. 1) is copied from a host processing system (e.g., host processor 102 and/or host memory 110 in FIG. 1) to a network protocol offload engine (e.g., TOE 118 in FIG. 1). Message information is provided to the network protocol offload engine. The

message information contains a message buffer location (e.g., address of memory fragments 112 containing packet payloads) in a host memory (e.g., host memory 110 in FIG. 1). One or more message contexts (e.g., message contexts 124 in FIG. 1) are generated in the offload engine from the message information to indicate the message buffer location (e.g., address of memory fragments 112) rather than copying the message buffer to the offload engine. Protocol processing is performed at the offload engine while leaving the message buffer (e.g., memory fragments 112 in FIG. 2) in the host memory (e.g., see para. [0012], [0023], and [0024]). Finally, the message buffer is transmitted in the form of at least one packet payload (e.g., segments payloads 204 in FIG. 2) directly from the host memory to a network communication link (e.g., network communication link 121 in FIG. 1 via MAC/PHY 120 in FIG. 1 or FIG. 2), without intermediate buffering of the at least one packet payload by the offload engine, during transmission of packets by the offload engine (e.g., see para. [0019]).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The issues presented in this appeal are:

(i) Whether independent claims 1 and 17 are obvious in view of Boucher and Ziai under 35 USC § 103(a).

(ii) Whether independent claim 9 is obvious in view of Boucher and Ziai under 35 USC § 103(a).

(iii) Whether dependent claims 2-5, 8, 20, and 21 are obvious in view of Boucher and Ziai under 35 USC § 103(a).

(iv) Whether dependent claims 10, 12, and 13 are obvious in view of Boucher and Ziai under 35 USC § 103(a).

(v) Whether dependent claims 6, 7, 18, and 19 are obvious in view of Boucher and the allegedly admitted prior art under 35 U.S.C. § 103(a).

VII. ARGUMENTS

This section sets forth Appellants' arguments against the rejections and in favor of the patentability of the claims on appeal. Part VII.A provides a brief overview of the Boucher and Ziai references, which the Final Office Action relied upon to reject the claims on appeal. Part VII.B discusses why the combination of Boucher and Ziai cannot render independent claims 1 and 17 obvious. Part VII.C discusses why the combination of Boucher and Ziai cannot render independent claim 9 obvious. Part VII.D discusses why the combination of Boucher and Ziai cannot render dependent claims 2-5, 8, 20, and 21 obvious. Part VII.E discusses why the combination of Boucher and Ziai cannot render dependent claims 10, 12, and 13 obvious. Part VII.F discusses why the combination of Boucher and the allegedly admitted prior art cannot render dependent claims 6, 7, 18, and 19 obvious.

A. Brief Overview of Boucher and Ziai

1. Boucher

Boucher discloses an intelligent network interface card (INIC) or communication processing device (CPD) 30 (see FIG. 2) which includes its own processor 55 (see FIG. 5) that frees the host CPU 28 (see FIG. 1) from most protocol processing. *Boucher*, para. [0035]. Fast-path processing enables CPD 30 to by-pass the host protocol stack 44 via fast-path 58 when transmitting and receiving packets (see FIG. 2). *Boucher*, para. [0037]. When a connection is determined to be a candidate for fast-path processing by CPD 30, the first frame in that connection is sent to the host CPU for slow-path processing through the host protocol stack 44 executed by the host CPU 28. The host 20 uses this first frame to generate a communication control block (CCB) for the connection. Host 20 then transfers the CCB to CPD 30, which buffers the CCB in its CCB cache 62. *Boucher*, para. [0043]. The CCB is a connection context defining the particular connection and includes information such as protocol type and source and destination addresses for each protocol layer. *Boucher*, para. [0036]. The CPD 30 uses the CCB to collapse the host protocol stack 44 into a signal state machine for fast-path processing. *Boucher*, para.

[0038].

With reference to FIG. 5, during fast-path processing to transmit packets, Boucher states,

> Guided by the CCB, the **processor 55 moves network frame-sized portions of the data from the source in host memory 35 into its own memory 60** using DMA, as depicted by arrow 99. The processor 55 **then** prepends appropriate headers and checksums to the data portions, and transmits the resulting frames to the network 25, consistent with the restrictions of the associated protocols.

Boucher, para. [0049] (emphasis added). This portion of Boucher discloses that CPD 30 moves frame-sized portions of data (i.e., message M 90 illustrated in FIG. 5) from host memory 35 into its own memory 60 and then prepends the appropriate headers and checksums. In other words, CPD 30 performs its processing with reference to a CCB for the particular connection stored in CCB cache 62 **after** the packet payload or message data is transferred from host memory 35 into memory 60 of CPD 30.

2. Ziai

Ziai discloses a method and apparatus for accelerating Internet Protocol Security (IPSec) processing. FIG. 3A illustrates an architecture for offloading IPSec processing from a host 306 to a network protocol offload chip 300 while FIG. 3C illustrates a method for outbound IP packet processing. *Ziai*, col. 4, lines 8-13. With reference to FIG. 3C, in block 303C, an IP packet is transferred from network offload memory (NOM) 304 or system memory 307 into crypto accelerator 302 within network protocol offload chip 300. Subsequently, in block 305C, IPSec processing is performed. In addition, Ziai states,

> Referring to FIGS. 3a and 3c, outbound IP packets begin as application data within the system memory 307. The **application data is then transferred (e.g. via Direct Memory Access) 301c by the system CPU/chipset 306 to the TCP/IP processors 305** (e.g., by writing the application data into the NOM 304 from the CPU/chipset 306 and reading the application data from the NOM 304 by the TCP/IP processors 305). **In alternate embodiments the application data may be written into the outbound network interface 301. Then, the TCP/IP processors 305 perform TCP/IP processing 302c on the application data**. In the outbound direction, TCP/IP processing is the addition of the TCP header 102 at the

transport layer 109 and the addition of the first IP header 103 at the network layer 110, consistent with the TCP and IP protocols. **In the outbound direction, TCP/IP processing results in the creation of an IP packet**. In addition, the IP packet is checked to determine if IPSec processing is required. Thus, after TCP/IP processing, an IP packet is stored in NOM 304. In alternate embodiments the IP packet may be sent to the outbound network interface 301.

Ziai, col. 5, lines 4-23 (emphasis added). This portion of Ziai teaches that an IP packet is not created until TCP/IP processing is performed and further that TCP/IP processing is not performed until after application data is transferred from system memory 307 into network protocol offload chip 300. Even in the disclosed "alternative embodiment" where the application data is buffered in network interface 301, Ziai still discloses that TCP/IP processing to create an IP packet is not performed until after application data is transferred into network protocol offload chip 300.

B. The combination of Boucher and Ziai do not render independent claims 1 and 17 obvious

Claims 1 and 17 stand rejected under 35 USC § 103(a) as being obvious over the combination of Boucher and Ziai.

"To establish prima facie obviousness of a claimed invention, **all the claim limitations** must be taught or suggested by the prior art. All words in a claim must be considered in judging the patentability of that claim against the prior art." M.P.E.P. § 2143.03.

Independent claim 1 recites, in pertinent part,

> **performing protocol processing at the offload engine while leaving the message buffer in the host memory**; and
> transmitting the message buffer in the form of at least one packet payload directly from the host memory to a network communication link, without intermediate buffering of the at least one packet payload by the offload engine, during transmission of packets by the offload engine.

Appellants respectfully submit that the combination of Boucher and Ziai fails to teach or suggest performing protocol processing at the offload engine while leaving the message buffer in the host memory.

The Final Office Action cites INIC/CPD 30 illustrated in FIG. 2 of Boucher as

corresponding to the claimed "offload engine" (which includes processor 55 as illustrated in FIG. 5) and cites the communication control block (CCB) as corresponding to the claimed message contexts. *Final Office Action* mailed 5/23/08, page 2. However, Boucher discloses,

> Guided by the CCB, the **processor 55 moves network frame-sized portions of the data from the source in host memory 35 into its own memory 60** using DMA, as depicted by arrow 99. The processor 55 **then** prepends appropriate headers and checksums to the data portions, and transmits the resulting frames to the network 25, consistent with the restrictions of the associated protocols.

Boucher, para. [0049] (emphasis added). This portion of Boucher discloses that CPD 30 moves frame-sized portions of data (i.e., message M 90 illustrated in FIG. 5) from host memory 35 into its own memory 60 and then prepends the appropriate headers and checksums. In other words, CPD 30 performs its processing with reference to a CCB for the particular connection stored in CCB cache 62 **after** the packet payload or message data is transferred from host memory 35 into memory 60 of CPD 30.

Furthermore, although CCB's are pre-generated prior to the transmission of packets in a connection subsequent to the first packet, the CCB is generated by the host CPU 28 executing the host protocol stack 44 on the first packet. Accordingly, the processing associated with creating the CCB does not teach or suggest performing protocol processing **at the offload engine** while leaving the message buffer in the host memory, since this processing is occurring not in CPD 30, but rather in host 20. Correspondingly, CPD 30 does not use the CCB to create headers and checksums until **after** packet payload data is transferred into CPD 30 itself.

In short, Boucher fails to teach or suggest performing protocol processing within CPD 30 while leaving the frame-sized portions of the data in host memory 35. Thus, Boucher fails to teach or suggest performing protocol processing at an offload engine **while leaving** the message buffer in host memory.

Similarly, Ziai also fails to teach or suggest the very same element. In fact, Ziai discloses,

> Referring to FIGS. 3a and 3c, outbound IP packets begin as application data within the system memory 307. The **application data is then transferred (e.g. via Direct Memory Access) 301c by the system**

> **CPU/chipset 306 to the TCP/IP processors 305** (e.g., by writing the application data into the NOM 304 from the CPU/chipset 306 and reading the application data from the NOM 304 by the TCP/IP processors 305). **In alternate embodiments the application data may be written into the outbound network interface 301. Then, the TCP/IP processors 305 perform TCP/IP processing 302c on the application data**. In the outbound direction, TCP/IP processing is the addition of the TCP header 102 at the transport layer 109 and the addition of the first IP header 103 at the network layer 110, consistent with the TCP and IP protocols. **In the outbound direction, TCP/IP processing results in the creation of an IP packet**. In addition, the IP packet is checked to determine if IPSec processing is required. Thus, after TCP/IP processing, an IP packet is stored in NOM 304. In alternate embodiments the IP packet may be sent to the outbound network interface 301.

Ziai, col. 5, lines 4-23 (emphasis added). This portion of Ziai teaches that an IP packet is not created until TCP/IP processing is performed and further that TCP/IP processing is not performed until after application data is transferred from system memory 307 into network protocol offload chip 300. Even in the disclosed "alternative embodiment" where the application data is buffered in network interface 301, Ziai still discloses that TCP/IP processing to create an IP packet is not performed until after application data is transferred into network protocol offload chip 300. Consequently, Ziai also fails to teach or suggest performing protocol processing at an offload engine **while leaving** the message buffer in host memory, where the message buffer stores the "packet payload."

Consequently, the combination of Boucher and Ziai fails to teach or suggest all elements of claim 1, as required under M.P.E.P. § 2143.03. **Independent claim 17 includes similar nonobvious elements as independent claim 1.** Accordingly, Appellants request that the instant §103(a) rejections of claims 1 and 17 be withdrawn.

C. The combination of Boucher and Ziai do not render independent claim 9 obvious

Claim 9 stands rejected under 35 USC § 103(a) as being obvious over the combination of Boucher and Ziai.

"To establish prima facie obviousness of a claimed invention, **all the claim limitations** must be taught or suggested by the prior art. All words in a claim must be

considered in judging the patentability of that claim against the prior art." M.P.E.P. § 2143.03.

Independent claim 9 recites, in pertinent part,

> an **engine to perform protocol processing** with information from the transmission control protocol block and additional information concerning a location of the packet payloads in the host buffer, the engine to create and to send packets on the communication link **according to the protocol processing** with the information from the transmission control protocol block and the additional information concerning the location of the **packet payloads in the host buffer while leaving the packet payloads in the host buffer**, the packet payloads being directly copied from the host buffer to the communication link, without intermediate buffering of the packet payloads within the engine, to complete packet transmissions.

Appellants respectfully submit that the combination of Boucher and Ziai fails to teach or suggest an engine to performing protocol processing while leaving packet payloads in a host buffer.

The Final Office Action cites INIC/CPD 30 illustrated in FIG. 2 of Boucher as corresponding to the claimed "offload engine" (which includes processor 55 as illustrated in FIG. 5) and cites the communication control block (CCB) as corresponding to the claimed message contexts. *Final Office Action* mailed 5/23/08, page 2. However, Boucher discloses,

> Guided by the CCB, the **processor 55 moves network frame-sized portions of the data from the source in host memory 35 into its own memory 60** using DMA, as depicted by arrow 99. The processor 55 **then** prepends appropriate headers and checksums to the data portions, and transmits the resulting frames to the network 25, consistent with the restrictions of the associated protocols.

Boucher, para. [0049] (emphasis added). This portion of Boucher discloses that CPD 30 moves frame-sized portions of data (i.e., message M 90 illustrated in FIG. 5) from host memory 35 into its own memory 60 and then prepends the appropriate headers and checksums. In other words, CPD 30 performs its processing with reference to a CCB for the particular connection stored in CCB cache 62 **after** the packet payload or message data is transferred from host memory 35 into memory 60 of CPD 30.

Furthermore, although CCB's are pre-generated prior to the transmission of packets in a connection subsequent to the first packet, the **CCB is generated by the host**

CPU 28 executing the host protocol stack 44 on the first packet. Accordingly, the processing associated with creating the CCB does not teach or suggest performing protocol processing **at the offload engine** while leaving the message buffer in the host memory, since this processing is occurring not in CPD 30, but rather in host 20. Correspondingly, CPD 30 does not use the CCB to create headers and checksums until **after** packet payload data is transferred into CPD 30 itself.

In short, Boucher fails to teach or suggest performing protocol processing within CPD 30 while leaving the frame-sized portions of the data in host memory 35. Thus, Boucher fails to teach or suggest an engine to performing protocol processing while leaving packet payloads in a host buffer.

Similarly, Ziai also fails to teach or suggest the very same element. In fact, Ziai discloses,

> Referring to FIGS. 3a and 3c, outbound IP packets begin as application data within the system memory 307. The **application data is then transferred (e.g. via Direct Memory Access) 301c by the system CPU/chipset 306 to the TCP/IP processors 305** (e.g., by writing the application data into the NOM 304 from the CPU/chipset 306 and reading the application data from the NOM 304 by the TCP/IP processors 305). **In alternate embodiments the application data may be written into the outbound network interface 301. Then, the TCP/IP processors 305 perform TCP/IP processing 302c on the application data**. In the outbound direction, TCP/IP processing is the addition of the TCP header 102 at the transport layer 109 and the addition of the first IP header 103 at the network layer 110, consistent with the TCP and IP protocols. **In the outbound direction, TCP/IP processing results in the creation of an IP packet**. In addition, the IP packet is checked to determine if IPSec processing is required. Thus, after TCP/IP processing, an IP packet is stored in NOM 304. In alternate embodiments the IP packet may be sent to the outbound network interface 301.

Ziai, col. 5, lines 4-23. This portion of Ziai teaches that an IP packet is not created until TCP/IP processing is performed and further that TCP/IP processing is not performed until after application data is transferred from system memory 307 into network protocol offload chip 300. Even in the disclosed "alternative embodiment" where the application data is buffered in network interface 301, Ziai still discloses that TCP/IP processing to create an IP packet is not performed until after application data is transferred into network protocol offload chip 300. Consequently, Ziai also fails to teach or suggest an

engine to performing protocol processing while leaving packet payloads in a host buffer.

Consequently, the combination of Boucher and Ziai fails to teach or suggest all elements of claim 9, as required under M.P.E.P. § 2143.03. Accordingly, Appellants request that the instant §103(a) rejection of claim 9 be withdrawn.

D. The combination of Boucher and Ziai do not render dependent claims 2-5, 8, 20, and 21 obvious

As to dependent claims 2-5, 8, 20, and 21, if an independent claim is allowable, then any claim depending therefrom is also allowable. *See, e.g.,* MPEP § 2143.03; *In re Fine*, 837 F.2d 1071 (Fed. Cir. 1988). As discussed above, independent claims 1 and 17 are in condition for allowance. Appellants submit that claims 2-5, 8, 20, and 21 are therefore allowable by virtue of their dependence on allowable independent claims, as well as by virtue of the features recited in the claims. Appellants respectfully request withdrawal of the rejections and allowance of these claims.

E. The combination of Boucher and Ziai do not render dependent claims 10, 12, and 13 obvious

As to dependent claims 10, 12, and 13, if an independent claim is allowable, then any claim depending therefrom is also allowable. *See, e.g.,* MPEP § 2143.03; *In re Fine*, 837 F.2d 1071 (Fed. Cir. 1988). As discussed above, independent claim 9 is in condition for allowance. Appellants submit that claims 10, 12, and 13 are therefore allowable by virtue of their dependence on allowable independent claim 9, as well as by virtue of the features recited in the claims. Appellants respectfully request withdrawal of the rejections and allowance of these claims.

F. The combination of Boucher and the allegedly admitted prior art do not render dependent claims 6, 7, 18, and 19 obvious

As to dependent claims 6, 7, 18, and 19, if an independent claim is allowable, then any claim depending therefrom is also allowable. *See, e.g.,* MPEP § 2143.03; *In re Fine*, 837 F.2d 1071 (Fed. Cir. 1988). As discussed above in section VII.B, independent claims 1 and 17 are in condition for allowance. The allegedly admitted prior art does not

address the identified deficiencies in Boucher and Ziai with respect to independent claims 1 and 17. Therefore dependent claims 6, 7, 18, and 19 are allowable by virtue of their dependence on allowable independent claims, as well as by virtue of the features recited in these dependent claims. Appellants respectfully request withdrawal of the rejections and allowance of these claims.

VIII. CONCLUSION

Given the above arguments supporting patentability, Appellants believes all claims on appeal are in condition for allowance. If the undersigned attorney has overlooked a teaching in any of the cited references that is relevant to allowance of the claims, the Examiner is requested to specifically point out where such teaching may be found. Further, if there are any informalities or questions that can be addressed via telephone, the Examiner is encouraged to contact the undersigned attorney at (206) 292-8600.

Charge Deposit Account

Please charge our Deposit Account No. 02-2666 for any additional fee(s) that may be due in this matter, and please credit the same deposit account for any overpayment.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: October 21, 2008

/Cory G. Claassen/
Cory G. Claassen
Attorney for Appellant(s)
Registration No. 50,296
Phone: (206) 292-8600

APPENDIX A — CLAIMS

1. (Previously Presented) A method of transmitting packets comprising:

copying a protocol control block from a host processing system to a network protocol offload engine;

providing message information to the network protocol offload engine, the message information containing a message buffer location in a host memory;

generating one or more message contexts in the offload engine from the message information to indicate the message buffer location rather than copying the message buffer to the offload engine;

performing protocol processing at the offload engine while leaving the message buffer in the host memory; and

transmitting the message buffer in the form of at least one packet payload directly from the host memory to a network communication link, without intermediate buffering of the at least one packet payload by the offload engine, during transmission of packets by the offload engine.

2. (Original) The method of claim 1 wherein said transmitting the message buffer comprises retrieving the message buffer from the host memory via cut-through transmissions.

3. (Original) The method of claim 2 wherein said cut-through transmissions comprise direct memory access copies.

4. (Original) The method of claim 1 wherein said performing protocol processing comprises processing TCP segments.

5. (Original) The method of claim 4 wherein said performing protocol processing comprises generating TCP headers for the TCP segments.

6. (Original) The method of claim 1 further comprising freeing the one or more message contexts upon acknowledgement of the packet payload delivery.

7. (Original) The method of claim 1 further comprising providing message completion information to the host processing system to release message buffers containing the packet payload.

8. (Original) The method of claim 1 wherein said performing protocol processing comprises processing machine-readable instructions stored in a storage medium.

9. (Previously Presented) A network offload engine comprising:

a first interface to a host processor to receive a copy of a transmission control protocol block;

a second interface to a communication link to copy packet payloads from a host buffer onto the communication link; and

an engine to perform protocol processing with information from the transmission control protocol block and additional information concerning a location of the packet payloads in the host buffer, the engine to create and to send packets on the communication link according to the protocol processing with the information from the transmission control protocol block and the additional information concerning the location of the packet payloads in the host buffer while leaving the packet payloads in the host buffer, the packet payloads being directly copied from the host buffer to the communication link, without intermediate buffering of the packet payloads within the engine, to complete packet transmissions.

10. (Original) The network offload engine of claim 9 wherein the additional information concerning the location of the packet payloads in the host buffer comprises at least one message context.

11. (Original) The network offload engine of claim 9 wherein the communication link comprises unshielded twisted pair wire for Ethernet communications.

12. (Original) The network offload engine of claim 9 wherein the direct copy of the packet payloads from the host buffer comprises a cut-through transmission of the packet

payloads to the communication link of the network offload engine.

13. (Original) The network offload engine of claim 12 wherein the copy of the packet payloads from the host buffer comprises a direct memory access engine to copy the packet payloads from the host buffer.

14. - 16. (Cancelled)

17. (Previously Presented) An article comprising:

a storage medium comprising machine-readable instructions stored thereon to: perform protocol processing at an offload engine while leaving a packet payload of a packet in a host memory;

access one or more message contexts that contain the packet payload address from the host memory to complete the protocol processing; and

transmit the packet payload directly from the host memory to a communication link, without intermediate buffering of the packet payload within the offload engine, during transmission of the packets by the offload engine.

18. (Original) The article of claim 17 wherein the storage medium further comprises machine-readable instructions to free message contexts upon receiving an acknowledgement of payload delivery.

19. (Original) The article of claim 18 wherein the storage medium further comprises machine-readable instructions to instruct the host processing system to release message buffers of the host memory upon receiving the acknowledgement of payload delivery.

20. (Original) The article of claim 17 wherein the storage medium further comprises machine-readable instructions to instruct the host processing system to perform protocol processing for TCP segments.

21. (Original) The article of claim 20 wherein the storage medium further comprises machine-readable instructions to instruct the host processing system to generate a TCP header for the TCP segments.

APPENDIX B— EVIDENCE

[No Evidence is Entered]

APPENDIX C — RELATED PROCEEDINGS

[No Related Proceedings]